```python
import heapq
graph = {
    'A': [('B', 1), ('C', 4)],
    'B': [('D', 3), ('E', 5)],
    'C': [('F', 2)],
    'D': [('F', 1), ('E', 1)],
    'E': [('F', 2)],
    'F': []
}
heuristic = {
    'A': 4,
    'B': 3,
    'C': 2,
    'D': 1,
    'E': 1,
    'F': 0
}
```

```
Node Expansion Order: ['A', 'B', 'D', 'F']
Optimal Path: A → B → D → F
Total Path Cost: 5

=== Code Execution Successful ===
```

main.py

```
def a_star(start, goal):
    pq = []
    heapq.heappush(pq, (heuristic[start], 0, start, [start]))
    visited = set()
    expansion_order = []
    while pq:
        f, g, node, path = heapq.heappop(pq)
        if node in visited:
            continue
        expansion_order.append(node)
        visited.add(node)
        if node == goal:
            return expansion_order, path, g
        for neighbor, cost in graph[node]:
            if neighbor not in visited:
                g_new = g + cost
                f_new = g_new + heuristic[neighbor]
```

>

```
                heapq.heappush(pq, (f_new, g_new, neighbor, path +
                    [neighbor]))
    return None, None, None
start_node = 'A'
goal_node = 'F'
expansion_order, optimal_path, total_cost = a_star(start_node, goal_node
    )
print("Node Expansion Order:", expansion_order)
print("Optimal Path:", " → ".join(optimal_path))
print("Total Path Cost:", total_cost)

```